Design of a Chatbot Social Engineering Victim

Ryan M. Schuetzler
Brigham Young University
ryan.schuetzler@byu.edu

Justin Scott Giboney
Brigham Young University
justin_giboney@byu.edu

G. Mark Grimes
University of Houston
gmgrimes@bauer.uh.edu

Abstract

Social engineering is an ever-growing problem in online and offline communication. Companies invest time and resources to train employees not to fall victim to attacks. The concept of adversarial thinking encourages people to learn the ways of the attacker to better defend themselves. This research introduces the design features of a chatbot that plays the role of a social engineering victim to allow people to perform the role of an attacker in a training exercise. By attacking this chatbot, people can learn better how to defend themselves.

1. Introduction

Social Engineering (SE), in its broadest definition, is the act of "skillfully maneuvering human beings to take action in some aspect of their lives" [1, p. 10]. When used in a cybersecurity context, SE refers to an attack vector that takes advantage of human habits and biases to gain unauthorized access to resources or information. Common social engineering techniques include phishing (the use of SE techniques over email *en masse*), shoulder surfing (peering over someone's shoulder to obtain confidential information from the screen), vishing (calling and pretending to be someone else to get information), smishing (a phishing attack conducted over SMS messages), and spear phishing (targeted phishing). SE techniques are employed to get access to confidential information or systems from businesses, organizations, or individuals.

Social engineering is a major security problem for many organizations. The Verizon Data Breach Investigation Report identified 3,841 SE incidents in 2020 [2]. SE breaches cost companies and countries billions of dollars every year [3]. As a result, companies dedicate resources developing policies and training employees to comply with those policies.

The challenge of SE is that it circumvents technical security measures and exploits human decision-making to gather information [4]. Building an effective defense against SE in an organization frequently involves training and development of effective policies to encourage employees to adhere to good security practices. For example, many organizations conduct internal phishing attacks to evaluate employee's understanding of good security practices [5].

Companies struggle with providing SE training that employees find interesting [6]. Researchers have techniques to make engaging training through mindfulness [7], embedded training [8], competition [9], just-in-time feedback [9], adversarial thinking [authors], and role-play [10]. While some of the technology that has been produced has achieved decent engagement ratings from participants, there is still room to improve.

Additionally, most of the attention has been focused on emails and phishing. With the growing interest in chat customer support services, there is need to have SE training in other contexts that employees find interesting. There is growing interest in training from the perspective of the attacker, called adversarial thinking [11]. Very little research on adversarial training has been produced, and even less in the SE context.

Because of the social nature of SE attacks, chatbots have a unique advantage compared to other technologies: they use natural language and elicit social responses from users [12]. We have developed a chatbot to perform a role in training to give users direct experience with the SE concepts they learn. We want the chatbot to engage users and teach effectively so learners can recognize SE techniques in the future and avoid becoming victims of an attack. As part of this development, we propose the following research question:

RQ1: What are the design features of a chatbot necessary to provide engaging and adversarial training?

As we work to develop a chatbot-enabled training on social engineering, we need to examine its efficacy compared to traditional methods. While chatbots are increasingly used to teach in various contexts (e.g., [13]), their efficacy is still an area ripe for study. It may also be that the social capabilities of a chatbot have detrimental effects on desired outcomes [14]. We therefore seek to study this second research question:

URI: https://hdl.handle.net/10125/79921
978-0-9981331-5-7

RQ2: Does a chatbot-enabled SE training provide better learning outcomes compared to traditional methods of training?

We use a design science methodology [15]–[17] to answer our research question. The design science methodology has six steps: (1) identify the problem to explore; (2) define the objectives of a solution; (3) design and develop an artifact as a possible implementation of our solution criteria; (4) demonstrate the use of the artifact; (5) evaluate the potential value of the artifact; and (6) communicate the design and significance of the artifact and findings [17]. We will describe the development of a chatbot capable of producing conversations where participants can pretend to be an attacker trying to obtain a password through SE.

2. Social Engineering Training

Much of the training around social engineering in companies is defensive in nature c.f., [7]–[10]. Employees are taught what social engineering is, the potential damage it can cause, and the risks to their own employment if they fall victim to such an attack.

Some research in the cybersecurity realm has demonstrated the effectiveness of *adversarial thinking* to help train security professionals to think like the attacker. Adversarial thinking involves shifting perspective to think like an attacker would. Adopting this mindset can help users better understand what attacks might come, how they might come, and what defenses they can employ. We designed our chatbot to give users an adversarial experience with SE, so they can better understand how SE attacks are conducted. With this in mind, we aim to increase adversarial thinking in those who use the SE Victim.

Traditional SE training consists of videos, online modules, and/or quizzes to test understanding. Typically, this training focuses on compliance with policies, restricting access to facilities, locking computers, or on phishing, one of the most common SE attack vectors. These trainings are often limited in their ability to engage learners due to a lack of interactivity. To stage an interactive SE training, organizations run into issues of scalability, as these would typically involve live training where a few people attempt social engineering tactics in front of the class, or perhaps role play with each other. But the social nature of the attacks means that usually at least two people are required for the conversation. Chatbots provide a way to have a conversation with only one person, enhancing the application of SE training so that everyone can try out the tactics they are learning about. Our goal is not to create excellent social engineers who can later carry out attacks. Rather, our goal is to increase understanding of SE through the application of techniques with a simulated SE victim.

With a design science methodology, after defining the problem to explore, we need to define the objectives of the solution. There are three objectives that an adversarial chat-based SE training needs to accomplish:

1. Allow the user to practice various SE techniques,
2. Provide the user feedback on what works and what does not work, and
3. Create a realistic environment, i.e., make it feel like there is a person on the other end.

The following sections will first explain the general engine of the chatbot, second, the scenario that allows users to use SE techniques, third, how the chatbot provides feedback to the user by making decisions, and fourth, how the chatbot engages in natural conversation.

3. Scenario & Conversation Flow

In the current study, participants interacting with the victim chatbot are provided with a scenario to practice SE techniques using RASA, a popular chatbot software. In the scenario, they are told to act as the assistant to Lina Cassler, the CTO of RipTech, a fictional company. In a chat interaction with tech support at their company, they are to try to convince Janet, the tech support agent, to provide them with the password to Lina's account.

The flow of conversation is presented in Figure 1 and is described here. When users first arrive at the chat page, they enter a message to begin the conversation. The chatbot introduces itself as Janet, the technical support agent. Janet uses a robot graphic as the avatar, and we make no attempt to deceive users about the nature of their conversation with a bot.

Once the user and chatbot have introduced themselves, the users can begin to attempt their social engineering to get the password. Each message is evaluated to determine if it satisfies certain criteria.

The first branch of conversation is if the user is requesting the password. If the user never requests the password, then Janet will never give them the password, regardless of the social engineering techniques employed. If the password has been requested, we set a variable in Rasa to store that, and the chatbot will respond appropriately.

The next check is to determine if a social engineering technique is being used. These checks are performed in series, and a single message can be both a password request and an SE technique. For example, if the user said, "I really need to get Lina's password in the next 10 minutes or I'm going to get fired," that would be an example of the urgency technique combined with a password request. If the chatbot recognizes an SE technique, we decrease the resistance metric.

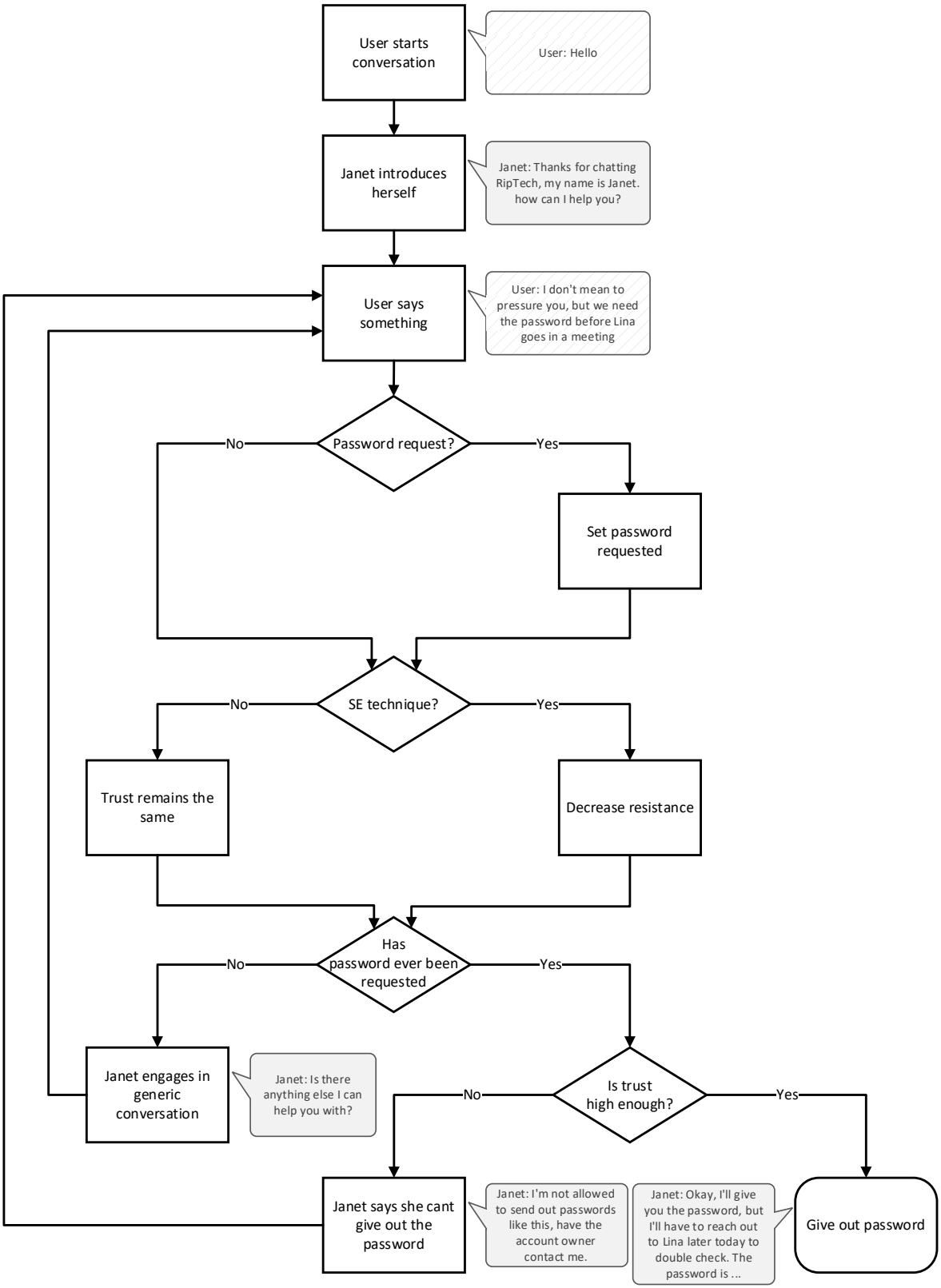

Figure 1. Flowchart of chatbot decisions

The crux of "passing" our social engineering training is through this resistance metric. In our current design, resistance is a singular metric to measure if users have adequately engineered the chatbot to convince it to provide the password. The Janet chatbot starts users with a resistance level of 50, and each social engineering attempt decreases that value by a random amount between 20 and 30. Once the resistance level reaches 0, the chatbot will give out the password (as long as it has been requested). These values were chosen based on intuition and will be excellent targets for future research. Ideas for future developments are discussed later in this paper.

At this point, dialog management takes over and Janet selects the response appropriate based on the state of conversation. If the user has requested the password and the resistance metric meets the threshold, the chatbot "reluctantly" provides the password in the chat. If the password has been requested, but resistance is not yet low enough, the chatbot responds that it cannot give the password at this time. The response is tied to the SE technique used in the last message. For example, if the user employed urgency as in the example above, Janet could respond with "I know you're in a rush and I want to help, but it's against policy to send out personal information." Finally, if no password has been requested, and resistance has not met the threshold, Janet will respond with a generic message to encourage further conversation.

4. SE Victim Architecture

There are many platforms for developing chatbots. One popular option is Rasa (https://rasa.com/)–an open source platform for developing machine learning-based chatbots. We chose Rasa for its flexibility and programmability. Rasa's platform allows for Python code to be integrated into the conversation cycle, allowing for powerful customization and capabilities. The development of a Rasa chatbot is composed of two major components: natural language understanding (NLU) and dialog management. NLU is the machine learning process of understanding user messages, parsing the meaning they are trying to convey, and extracting necessary information from those messages. Dialog management relates to how the chatbot determines what a user is doing in the conversation and how the chatbot's next message is chosen. Rasa provides both components to create a chatbot, though they can be used independently. The functionality of Rasa is similar to Google's DialogFlow or Amazon's Lex, with the added benefit of being open source and self-hosted, giving more control over the functionality and security of the data.

Rasa uses supervised machine learning to perform the natural language understanding (NLU) tasks. Using advanced modern language processing techniques, Rasa can understand a variety of speech patterns to create a natural flow of conversation with a modest amount of training. After evaluating other products, including DialogFlow and Microsoft's Bot Framework, we chose Rasa for its ease of use and customizability.

The primary building blocks of a Rasa bot are intents, responses, and stories. An intent is the user's presumed intent with an utterance. For example, if a user talking to an Echo device were to say "Alexa, set an alarm for 10:30," the intent could be named something like `set_alarm`. Separate intents would be needed for each of the possible tasks or types of messages users may want to send. Training data would then be used to provide the machine learning algorithm with many examples of how users might want to express the intent to set an alarm. For example:

- Set an alarm for 10:30
- I need an alarm set for 8:15
- Wake me up at 6:30
- Set an alarm

Each training phrase is unique and forms an input to the model that will eventually be used to classify the message by its intent.

In short, the Rasa framework provides natural language understanding to process user messages, dialog management to select appropriate responses, and connections to other platforms to integrate the chat. All of these are invaluable for creating a natural conversational flow.

The SE victim chatbot is designed to provide feedback to the user based on whether it gives away the password or not. The chatbot is designed to recognize different SE tactics. Chatbots use intents (or categories) to recognize user patterns. The intents we used centered on the primary categories of social engineering tactics used to get information: Urgency, threats, consensus, authority, reciprocity, and fear of missing out. For example, if the user decided to try a reciprocity approach, they might say something like the following:

- If you can get me this password, I can put in a good word with the boss.
- Listen, if you help me this one time, I can help you out
- Hey listen, I know you're up for a raise. Send me the password and I'll put in a good word.

When users employ different strategies, the NLU module classifies the intent and selects the appropriate response. Rasa's responses provide the chatbot with a variety of ways to reply to user messages. In Rasa, responses are created independently of the intents, so they can be reused or remixed to provide the right kind of conversation. The chatbot developer can introduce

variety to the responses so users are less likely to get repetitive messages from the chatbot. In the SE victim chatbot, we gave the chatbot several ways to deny users when they repeatedly asked for the password before employing social engineering techniques:

- I still can't just send you the password.
- Listen, I would love to help, but I really can't send it to you.
- I can't give you the password over this channel. Have the account owner call me.
- Like I said, I can't just give you the password info.

This variety mimics a human conversation, because a person chatting would not simply repeat the exact same message over and over again [12]. In our case, we have different responses based on the strategy the user is using to pressure the chatbot. For example, if the user tries to use reciprocity, the chatbot would respond with one of the following:

- I'd love the help, but I would get in trouble for sending personal info
- I really appreciate your interest, but I would have to have the account owner call me
- That's really thoughtful, but I really shouldn't send personal information to you

Stories are Rasa's way of controlling the flow of conversation. On their own, intents and responses are not linked, and provide no actual conversation. With stories, you can define a sequence of intents and responses that you expect users to follow. Intent detection understands what users want to do, and the responses guide them to the next step in the story. Multiple stories can be combined in a single conversation to handle the entire conversation and variety of ways people communicate.

Much of Rasa's strength comes from its customizability and programmability. Because the functions of Rasa are built with Python and provide bindings to allow programmatic access, the ways we can use Rasa are limited only by our programming ability. The crux of our social engineering victim chatbot is using Rasa's action server to power custom functionality. Creating custom actions gives us access to functions beyond the NLU itself. The chatbot's model of resistance is managed using a custom action. At the beginning of the interaction, the chatbot has a base level of "resistance" that users must decrease through their actions. In this case, resistance is an umbrella term covering a variety of responses to social engineering. The purpose is to force users to "wear down" the victim through repeating strategies or trying different strategies to get the information. The resistance metric decreases when users employ one of the SE tactics, although the users cannot see what their resistance level is. Once the resistance has passed below the defined threshold, the user can ask for the password and the chatbot will "reluctantly" respond with the password.

The purpose of the resistance metric is to mimic the way a social engineering attack might happen in the wild. Social engineering attacks are often based on the attacker developing rapport with the target. The tactics identified in social engineering research work to "soften up" the target and make them more likely to give up information that they should not. With the current SE victim, there is no way for users to increase resistance—it only decreases. Future iterations will also include the possibility of "locking themselves out" by arousing suspicion or employing tactics improperly.

Entity recognition in NLU is the process of identifying keywords or phrases that carry special significance. For example, when responding to “What is your name,” you might say “I’m Alice,” or “My name is Alice,” or one of many other ways you might introduce yourself. Entity recognition would be used to extract “Alice” as the name regardless of how the introduction was phrased.

Once the entity is extracted, it can be immediately used, or stored in a slot. Slots are like Rasa’s long-term memory. They can be used to store information for a short time, or for the duration of the conversation. In the example of the user’s name, a slot called `name` could be used to store the name, then future responses could incorporate the name (e.g., “Nice to see you again, Alice.”).

We use entity recognition to identify when users are referring to the CTO who was the target of the attack, and when users were asking for the password or login information. We use slots to remember if the requests had been made, and only if that information has been requested will the chatbot provide the password.

5. Testing, Evaluation, and Iteration

Testing the chatbot with users is a critical component of development. Just as with user testing in other research and development fields, conversations with actual users have no substitute. The Rasa team has developed the term “Conversation Driven Development” or CDD to refer to an iterative development process involving early and frequent interactions with users [18].

CDD is valuable for conversational agent development because natural language is so infinitely variable. There is no way for any person or group of people to predict all the ways another group will choose to interact with a chatbot. Rather than attempt to predict all possible conversation styles, CDD proposes that conversational tests with users be used repeatedly during the development process to identify weaknesses and to iteratively improve the chatbot using the

conversations as input for training machine learning models and shaping conversations.

The first test of our SE victim was conducted with a small class of cybersecurity students. After a class lecture on social engineering, students were given a link to the chatbot and brief instructions about the pretext associated with the chatbot (as described in Section 3 above). Then they were given free rein to interact with the bot, including the option to restart the conversation as many times as they wanted. Our main goal with these participants was to gather more data on how participants comfortable with concepts of social engineering would approach the bot.

Surprisingly, in our initial tests some users would begin their conversation and never indicate what exactly they needed from the chatbot. Because it is a contrived scenario, users must have assumed that the chatbot knew they wanted the password, so they began immediately employing social engineering without ever specifying their goal. This led to the addition of the slot for requesting the password mentioned above.

The second test of the SE victim was with a large class of information systems students enrolled in a security and controls class. These students had just learned about social engineering and were told to use the chatbot to apply the concepts they had learned. While their class training was defensive in nature, their attacks on the chatbot showed a great deal of creativity and insight. In addition to providing more training data for our existing intents, examining their conversations allowed us to improve the chatbot's conversation in several ways. While quantitative measures were not collected in this phase, qualitative assessment of student responses suggest the chatbot experience was positive. Additionally, many participants restarted the conversation multiple times, even after successfully obtaining the password, to see if they could obtain the password in different ways.

First, we used the pilot conversations to identify new intents to represent a variety of techniques we had not considered. One new intent was what we termed "minimizing," where the attackers attempt to convince the victim to share information saying things like "No one will know," or "It's not that big of a deal." In this way, the attackers hope to reduce the perceived impact of the security violation so that the victim will share the information.

Another new intent we saw from our test subjects was the "unreachable" intent. In our scenario, attackers were attempting to impersonate the assistant to the CTO of a fictional company. The chatbot would respond to early attempts to access the password by deflecting with requests to talk to the CTO herself. Several attackers attempted to circumvent this deflection by claiming that Lina (the CTO) was unreachable at the moment, but that the password needs were urgent. One participant even claimed to be Lina's husband, and that Lina had died and her accounts needed to be cleaned up.

Second, we identified the need for the chatbot to handle small talk. Especially with the social nature of a social engineering attack, a chatbot needs to be able to handle chit chat or banter that attackers use to attempt to build rapport with their target. Hobert and Berens [19] showed that small talk is important, especially in the early stages of a human-chatbot dialog, as users probe the capabilities of the chatbot. Here we saw similar probing behavior, and have enhanced the chatbot with an ability to respond to or deflect small talk.

6. Future Work

Our current tests have begun to show the feasibility of using the Rasa chatbot framework to create an SE victim for the purpose of training users. The studies conducted so far have focused on enhancing the chatbot's ability to respond to users and manage conversations in a natural way.

Future work will focus on experimental testing of the chatbot's ability to enhance engagement and learning compared to traditional SE training materials. We hope to show that through adversarial experience, even if simulated, learners get a better appreciation for how an attack might happen. We also hope to show that a chatbot that enables adversarial thinking improves engagement through its interactivity.

To properly compare the effectiveness of chatbot-enabled and traditional training, we must first develop a reliable measure of understanding of SE concepts. We are currently working to develop such a measure. Once established, we will conduct experiments to compare both theoretical and practical understanding of SE concepts after the two types of training.

In the long term, we also have plans to improve the chatbot to make the experience more realistic and valuable. One of these plans is to allow the resistance value to increase as well as decrease. Currently the chatbot only decreases its resistance to the SE attempts, resulting in giving out the password eventually as long as the user continues to try. In future iterations, the chatbot will contain an upper threshold for resistance, such that users could "lock" the encounter and the chatbot would refuse to respond or give the password at all. Some thoughts on how to lock the encounter:

- Threatening too intensely – threats of violence would be unlikely to lead to success in a normal social engineering attempt, especially over a text chat.
- Shotgun approach – trying too many different social engineering techniques.

We also plan to use multiple metrics besides just a single resistance metric to model the internal thought processes of a victim more accurately. For example, suspicion, trust, and liking are three different components of how a user approaches a conversation. Threats might reduce the resistance metric but increase the suspicion metric. We have not decided on specific details on how this process will be modeled, but we think it will lead to a more accurate representation of how social engineering attacks are carried out.

7. Conclusion

In summation we have built a chatbot that allows people to perform the role of an attacker to better learn how to defend themselves. This research introduced the design features of a chatbot that plays the role of a social engineering victim. Using this chatbot, people should be better prepared in the art of adversarial thinking and better able to defend themselves from social engineering attacks.

8. References

[1] C. Hadnagy, *Social Engineering: The Art of Human Hacking*. John Wiley & Sons, 2010.

[2] Verizon, "2021 Data Breach Investigations Report (DBIR)," 2021. Accessed: Jun. 15, 2021. [Online]. Available: https://enterprise.verizon.com/resources/reports/2021-data-breach-investigations-report.pdf

[3] F. Mouton, M. Teixeira, and T. Meyer, "Benchmarking a mobile implementation of the social engineering prevention training tool," in *2017 Information Security for South Africa (ISSA)*, Aug. 2017, pp. 106–116. doi: 10.1109/ISSA.2017.8251782.

[4] A. Abbasi, D. Dobolyi, A. Vance, and F. M. Zahedi, "The Phishing Funnel Model: A Design Artifact to Predict User Susceptibility to Phishing Websites," *Information Systems Research*, Feb. 2021, doi: 10.1287/isre.2020.0973.

[5] R. Wright and J. B. Thatcher, "Phishing Tests Are Necessary. But They Don't Need to Be Evil.," *Harvard Business Review*, Apr. 01, 2021. [Online]. Available: https://hbr.org/2021/04/phishing-tests-are-necessary-but-they-dont-need-to-be-evil

[6] A. J. Burns, M. E. Johnson, and D. D. Caputo, "Spear phishing in a barrel: Insights from a targeted phishing campaign," *Journal of Organizational Computing and Electronic Commerce*, vol. 29, no. 1, pp. 24–39, 2019.

[7] M. L. Jensen, M. Dinger, R. T. Wright, and J. B. Thatcher, "Training to Mitigate Phishing Attacks Using Mindfulness Techniques," *Journal of Management Information Systems*, vol. 34, no. 2, pp. 597–626, Apr. 2017, doi: 10.1080/07421222.2017.1334499.

[8] P. Kumaraguru *et al.*, "Getting users to pay attention to anti-phishing education: evaluation of retention and transfer," in *Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit*, New York, NY, USA, Oct. 2007, pp. 70–81. doi: 10.1145/1299015.1299022.

[9] S. Karumbaiah, R. T. Wright, A. Durcikova, and M. L. Jensen, "Phishing Training: A Preliminary Look at the Effects of Different Types of Training," in *Proceedings of the 11th Pre-ICIS Workshop on Information Security and Privacy*, Dublin, Ireland, 2016, p. 11.

[10] Z. A. Wen, Z. Lin, R. Chen, and E. Andersen, "What.Hack: Engaging Anti-Phishing Training Through a Role-playing Phishing Simulation Game," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, New York, NY, USA, May 2019, pp. 1–12. [Online]. Available: https://doi.org/10.1145/3290605.3300338

[11] J. Thompson *et al.*, "Student Misconceptions about Cybersecurity Concepts: Analysis of Think-Aloud Interviews," *Journal of Cybersecurity Education, Research and Practice*, vol. 2018, no. 1, Jul. 2018, [Online]. Available: https://digitalcommons.kennesaw.edu/jcerp/vol2018/iss1/5

[12] R. M. Schuetzler, G. M. Grimes, and J. S. Giboney, "The impact of chatbot conversational skill on engagement and perceived humanness," *Journal of Management Information Systems*, vol. 37, no. 3, pp. 875–900, Jul. 2020, doi: 10.1080/07421222.2020.1790204.

[13] J. Ceha, K. J. Lee, E. Nilsen, J. Goh, and E. Law, "Can a Humorous Conversational Agent Enhance Learning Experience and Outcomes?," in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, New York, NY, USA, May 2021, pp. 1–14. doi: 10.1145/3411764.3445068.

[14] R. M. Schuetzler, G. M. Grimes, J. S. Giboney, and H. K. Rosser, "Deciding Whether and How to Deploy Chatbots," *MIS Quarterly Executive*, vol. 20, no. 1, p. 16, 2021.

[15] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *MIS Quarterly*, vol. 28, pp. 75–105, 2004.

[16] J. F. Nunamaker and R. O. Briggs, "Toward a broader vision for information systems," *ACM Transactions on Management Information Systems*, vol. 2, 2011.

[17] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *Journal of management information systems*, vol. 24, no. 3, pp. 45–77, 2007.

[18] A. Nichol, "Conversation-Driven Development," *The Rasa Blog: Conversational AI Platform, Powered by Open Source*, May 07, 2020. https://blog.rasa.com/conversation-driven-development-a-better-approach-to-building-ai-assistants/

[19] S. Hobert and F. Berens, "Small Talk Conversations and the Long-Term Use of Chatbots in Educational Settings – Experiences from a Field Study," in *Chatbot Research and Design*, Cham, 2020, pp. 260–272. doi: 10.1007/978-3-030-39540-7_18.